Final Report Project Report Evaluation

Team Name: Weiner

Content	Exceptional 3	Satisfactory 2	Poor 1	Missing 0	Score
Abstract	✓	✓			2
Objective	✓				3
Theory		✓			2
Methods	✓				3
Results	✓				3
Conclusions & Recommendations	✓				3
Appendix	✓				3
Format	**Exceptional 1**	**Satisfactory .7**	**Poor .5**	**Missing 0**	**Score**
Punctuation, Grammar, Spelling, Clarity, Proofreading, Organization	✓				1
Figures, Tables, Graphics		✓			0.7
Cover Page/TOC	✓				1
Citations & References	✓				1
TOTAL					

19

3.7

22.7

Rose-Hulman Institute of Technology

ECE425-01

Introduction to Robotics

Final Project: Navigation Competencies & Wireless Communication

Team Weiner/ Asimo

[redacted]

02/18/2018

Abstract

The final project is a combination of knowledges we learned from this course including basic robot movements and sensor feedbacks. Using these concepts, the robot is able to perform the four required complicated tasks, topological path planning, metric map path planning, localization and mapping, for this project. The robot is able to move based on the navigation command for topological path planning, follow the path using wavefront algorithm for metric map path planning, locate itself in the world for localization and create its own map for mapping. For topological path planning, the robot is able to successfully execute the navigation commands; for metric map path planning, the robot was able to plan its own paths to destination in the worlds; for localization, the robot was able to localize itself at different starting locations; and the mapping function worked well except for some odometry error. Overall, the robot successfully achieve all tasks required in the final project.

no S

Good job, a little more detail would be helpful

Table of Contents

Objective

The final project is a combination of multiple concepts we learned from this course. There are four main tasks in the project, which are topological path planning, metric map path planning, localization and mapping. Topological path planning allows robot to move in the artificial world based on navigation commands and sensor feedbacks. Metric map path planning uses wavefront algorithm to plan a path for the robot to move from starting point to goal. Localization allows robot to determine its location in the world. Mapping allows robot to use sensor feedbacks to come up with its own map in an unknown world.

To achieve these tasks, we used the knowledge from the most basic robot movements, such as forward, reverse, and turning, to more complicated sensor related behaviors, such as wall following. All tasks will be achieved using wireless communication to allow the robot to send and receive data. By the end of this project, the robot is able to locate itself in the artificial world using sensor feedbacks and move itself to the given goal in the world using either topological or metric map path planning.

Theory

There are several theories that went into the completion of the various objectives in this lab. Some of the main theories involve representation, path planning, and mapping or coverage behavior.

We used two different representations through the different parts of the lab. We used a simple occupancy grid where the world was divided into cells and the cells with obstacles in them were represented by 99's entered in the cells of a matrix. Passable cells were represented by 1's entered in the cells. An example of an occupancy grid map is shown in **Figure 1** below.

99	1	99	1
99	1	1	1
1	1	99	1
1	99	99	1

Figure 1: 4X4 Ocupancy Grid Represenation

The second strategy that we used for mapping was topological mapping. In this strategy the "walls" of a cell were described allowing for walls between cells while still having both cells passable. This is more accurate and allowed for easier localization. The bits of a byte were used for each cell. The least significant 4 bits of each byte represented if there was a wall on a given side or not. Details of this strategy are shown below in **Figure 2**. The least significant bit represents the front wall, the next bit represents the left side. The third bit represents the rear wall, and finally the fourth bit represents the right wall.

0b0011	0b0001	0b1111	0b1011
0b0011	0b0010	0b1001	0b0001
0b0011	0b0011	0b0011	0b0011
0b0010	0b1111	0b1111	0b1110

Should this correlate with Figure 1? Is this a real world?

Figure 2: 4X4 Topological Map

The next task involved path planning. The method that we used for path planning was grass fire expansion. We used a occupancy grid to simplify the path planning programming. The algorithm starts at the destination and works back to the goal by calculating which passable cell is closest to the starting point. It then treats this cell as its current position and repeats. With a few iterations this allows the robot to plan a path back to its starting point. Our path planner only considers movement at 90 degrees, so from each cell there are four possible positions. It is possible to move at 45 degrees; this expands the number of cells to check

You did not discuss topological path planning

complicating the path planning and also increasing possibility for odometry error. With each movement the next cell is marked with a higher number; this way the robot knows what order to follow the path in. This path is tracked in an array in the code with zeros representing things not in the path and other digits representing the path with order. An example of a planned path is shown in **Figure 3** below.

Specify the start and goal in the text.

1	0	0	0
2	3	4	5
0	0	0	6
0	0	0	7

Figure 3: 4X4 Path Planning (Path starts in the lower right and moves to the upper left)

In order to cover the whole world for mapping we used a coverage algorithm that tried to move up a whole column at a time before progressing to the next. In order to do this the robot moves forward as many cells as possible before hitting a dead end and turning around and moving forward to make sure there are no cells the opposite direction in the column that have not been visited yet. After the column is complete, the robot moves on to the next column. This process is then repeated for each column until the map is complete. This provides coverage for the whole map.

A graphic would help with this.

You need more details on localization and mapping

Methods

There are four main tasks in the final project, which are topological path planning, metric map path planning, localization and mapping. To achieve these tasks, we also use wireless communication for the robot to send and receive sensor feedbacks data.

Hardware

For the final project, we added two nrF2401 wireless transceivers. One on the robot and one on a Arduino Uno board connected to the computer. Since the transceiver only takes pin 50, 51, 52 for the input pins, which are occupied by the stepper motor, we moved the stepper motor pins to 30, 31, and 32 to avoid conflicts.
We did not add any new sensors to the robot. The only sensors needed to achieve the tasks for the project are the front, left, right and back sensors. Before getting into the programming, we detached the photoresistors from the robot. We set up two LEDs to pin 41 and 42 on the robot. The green LED is to indicate whether the IR sensors are reading correctly. The green LED is used in mapping to indicate when a robot thinks it is in a new cell.

Topological Path Planning & Execution

The robot's goal for this part is to be able to make navigation decision at distinctive places. The distinctive places in the artificial world are hallways, corners, t-junctions, and dead ends. As the robot encounters each gateway, it should be able to to make decision based on the given navigation commands. There are four letters used for navigation command, which are S = Start, L = go Left, R = go Right, and T = Terminate. For example, the given command is "SLRT." The robot should start moving forward following the hallway until it encounters the first gateway. At the gateway, it will then make a decision on whether it should turn left or keep moving forward. If the gateway is located on the left side of the robot, the robot should follow the given commands and make a left turn then continue moving forward until it encounters the next gateway, where it will make a decision on the next navigation command. If the gateway is located on the right side of the robot, the robot will then continue moving forward until it encounters a gateway located on its left side. The robot will stop when it finally get to the T command.

Metric Map Path Planning & Execution

For this part of the project, we divided the 4 x 4 world into 16 blocks and use wavefront algorithm to give each block a value to identify its wall locations. We then create a path for the robot to move from its starting position to the goal point. There are two ways to create the map, which are using occupancy grid and topology. For occupancy grid, the free spaces in the world are represented by 0s while the occupied spaces are represented by 99s. To make this type of metric map work, set the number from large (starting point) to small (goal point) as the location gets closer to goal point. The goal for the robot is to move from large number to small number in a reducing pattern. For topology, there are a total of 16 (0 to 15) numbers representing each possible wall locations. Using the 4 digits binary, 0000 indicates no walls are found, 0001 indicates front wall only, 0010 indicates left wall, 0100 is back wall and

1000 is right wall. If there are both front and back walls, the binary will be written as 0101 and hexadecimal will be 5. Since 4 walls make up an occupied space, there are a total of 16 possible wall locations. Different from the occupancy grid, there are walls considered in the world instead of only occupied and free spaces. Navigation in involved in this type of map and robot uses behaviors such as forward, turning left/right, follow hallway, and avoid obstacle.

Wireless

The robot was set on the floor without any cords plugged in. The TX is connected to the computer and the RX is connected to the robot. We type the data needed to be sent to the robot on the command line of the serial monitor. We use Serial.read( )[1] function to read the data being typed into the serial monitor and store it in a variable called "data". While the robot is turned on, it should wirelessly receive the variable data from the computer response to the commands. This is accomplished with the function radio.write( )[2] on the transmitter side to write the data to the channel and radio.Read( ) on the receiver side to read the data written earlier on the same channel.

Localization

The robot was given a end goal position and a complete map of the world but no start position, it is required to plan its own path to the goal position starting from a random cell it is being placed inside the world. The map is saved as a matrix with each cell being represented by a binary number. To achieve localization, the robot needs to first identify which cell it is placed into and which direction is facing. The robot identify its cell position using the front IR, left IR, right IR, and back IR sensors. We first set a threshold of 150 for the front IR sensors and 250 for the left and right IR sensors. If the data output from the sensors are bigger than the thresholds, the robot will detect a wall and set the corresponding bite of the wall to a *1*[3]. If no wall is being detected, the corresponding bite of that wall will be set to 0. Once the bite number are put together to form a complete binary number, we search through the number to the 16 binary numbers representing each cell on the map we saved earlier. If there is a match using, the robot will set its current location to the position on the map. If there are multiple matches, the robot will move forward and repeat the process until there is only one match.

The next step for localization is to plan the robot's path to the next cell it needs to move to. To simplify the task, we made the robot to always start localization facing north. Every time the robot reach a new cell, it will spin to face north again before checking its current cell and decides what the next cell it needs to move to. To decide which cell it needs to go next, we create a solveMap( ) function that solve the optimum option for the robot to move next.

Mapping

The robot started with knowledge of its starting location; our robot always started at (3,3) to simplify mapping. The general plan was then to have a function that correctly marked obstacles on the map given only the robot's current position and IR sensors. Then there was a separate function that handled the robot motion and called the first function that updates the

map when it arrived at each new position. This move function moved the robot to each cell in a column and then progressed the robot to the starting position of the next column where it was called again. This process was then run four times because there are four columns in the map. With this strategy the mapping size could easily be expanded by increasing the number of iterations of the loop. At the end of the run, the map was printed to the screen via the serial terminal to indicate that the robot had successfully mapped the world.

Results

Topological Path Planning & Execution

The robot was able to successfully execute commands fed to it through the wireless link. When the robot received a set of commands it would look for distinctive places, and then execute the action dictated by the radio commands. The only small challenge was that occasionally the robot would get a bad IR sensor reading leading it to believe it was at a different distinctive place than it was. This could have been fixed by averaging several sensor values or using a couple of different kinds of sensors.

Metric Map Path Planning & Execution

Our metric path planning was very successful and was able to plan paths to destinations in two separate worlds. The only slight drawback was that there is a possibility of the robot getting into a stuck state solving the map. This could be solved as discussed earlier, but was not necessary for this assignment.

Wireless

Wireless communication was successful overall. We opted to use it only for the topological path planning due to time constraints, but given more time we could have incorporated wireless into the other tasks of this project.

Localization

The robot was successful at localization. It was able to localize many times from different starting locations. Our algorithms relied on the robot always starting facing one direction, but this could be solved with the addition of a magnetometer or other sensor to give the robot a sense of direction.

Mapping

Over all, the mapping function worked well with the exception of odometry error. The size of the cells did not leave much tolerance for the robot to turn around repeatedly and with error, so the robot often scraped the wall on turns if no assistance was provided. The robot was able to provide correct maps for both worlds in which it was tested with minimal human intervention to correct for odometry error.

Conclusions & Recommendations

The robot can overall achieve the goals of the project. It is able to locate its current position and plan its own path to reach the goal position. It is able to communicate with the computer using TX and RX and perform commands sent through serial monitor. However, the robot has large odometry error while running. It constantly require physical adjustment to avoid hitting the walls. We would recommend some wall following mechanisms for improvement of the odometry error. Since IR sensors are used to locate the robot in the world, we would recommend sonar sensors for wall following to avoid data interference.

Questions

1. Were there any issues with the wireless communication? How could you resolve them?

The largest issue with the wireless communication was sending more than one value. We were eventually able to overcome this with for loops sending one value at a time.

2. What does the state machine, subsumption architecture, flowchart, or pseudocode look like for the path planning, localization, and mapping?

To simplify the project and make it easier to debug, we separate each part into a seperate file instead of making a subsumption architecture combining all the tasks. The flowcharts are attached below in the appendix section.

3. How would you implement SLAM on the CEENBot given what you have learned about navigation competencies after completing the final project?

Topological mapping is one of the SLAM algorithms that we used for this project, it helps us to capture connectivity of the world rather than create a geometrically accurate map. It helps when we need to solve the map and plan the path of the robot. We also used "Exploration" which is another SLAM algorithm. The robot is able to decide where to move next in order to reach the goal as efficient as possible. This is accomplished by create a new map variable and assign numbers to the new map and the robot will move from a large number to zero in a descending order.

4. What was the strategy for implementing the wavefront algorithm?

We used a occupancy grid to simplify the path planning programming. The algorithm starts at the destination and works back to the goal by calculating which passable cell is closest to the starting point. It then treats this cell as its current position and repeats. With a few iterations this allows the robot to plan a path back to its starting point. Our path planner only considers movement at 90 degrees, so from each cell there are four possible positions. It is possible to move at 45 degrees; this expands the number of cells to check complicating the path planning and also increasing possibility for odometry error.

5. Were there any points during the navigation when the robot got stuck? If so, how did you extract the robot from that situation?

The robot never got stuck during navigation. If it did, we could add a timer that would change to robots direction after so much time to free it from a stuck position.

6. How long did it take for the robot to move from the start position to the goal?

The robot took about 45 seconds to move from a goal on one side of the map to the other (about 8 cells).

7. What type of algorithm did you use to selection the most optimal or efficient path?
The algorithm that we used started at the goal and tried to chose the next open position that was closest to the goal each time. By repeating this choice until we arrived at the goal, a complete path was formed. In this manner, we should have an optimal path to the goal.
8. How did you represent the robot's start and goal position at run time?
At run time, the goal is represented by a 6 on the map and the start is represented by a 3 on the map.
9. Do you have any recommendations for improving that robot's navigation or wavefront algorithm?
The algorithm could have been improved by having a method to detect stuck cases and to account for these. With our current program it is possible to have a map configuration that results in the robot not being able to get to the goal. This could be corrected by programming the robot to back up along the path until it could make another choice when the stuck state is detected.
10. How did you use the serial monitor and bi-directional wireless communication to represent the map?
We used the USB cable and serial prints to send the map back to the computer and print them to the serial monitor. The process for using the wireless would have been similar, but there would have been the radio link in the middle and the second Arduino connected to the computer would have executed the print statements to indicate the map on the computer.
11. What type of map did you create and why?
For localization we used a metric map because it was easier to identify features and therefore easier to know where we were. For the mapping we used an occupancy grid because it was simpler to tell where we could and couldn't travel.
12. What was key in the integration of the localization, mapping, and path planning?
The main key in integrating the functions was making sure that they were happening at the right time. For example the map had to be made before the path could be planned. The robot has to know where it is in order for the map to make sense.

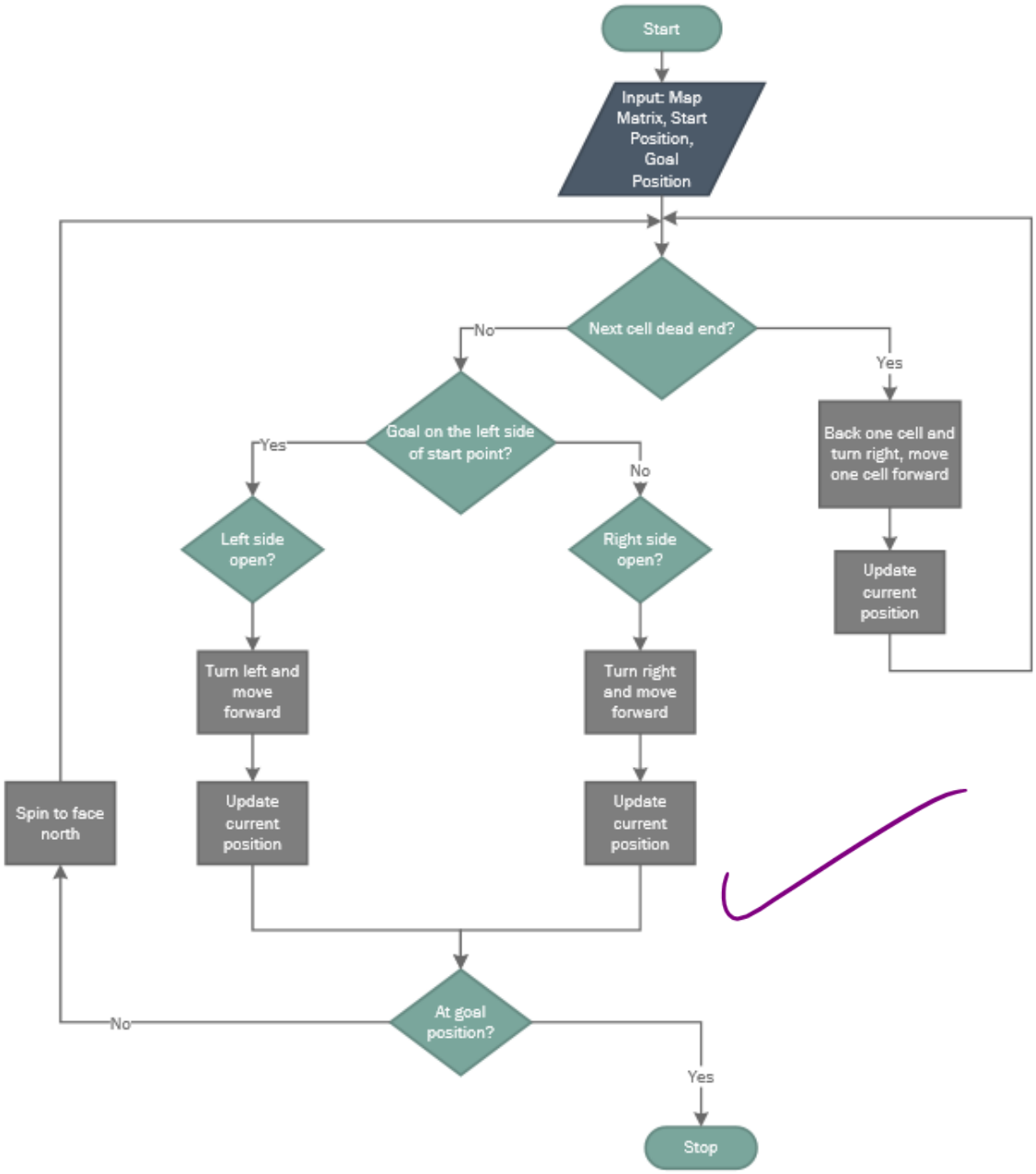
References

[1]*How to use Serial.read( ):* https://www.arduino.cc/en/Serial/Read
[2]*How to use Serial.write( ):* https://www.arduino.cc/en/Serial/Write
[3]*How to use bitWrite( ):* https://www.arduino.cc/reference/en/language/functions/bits-and-bytes/bitwrite/

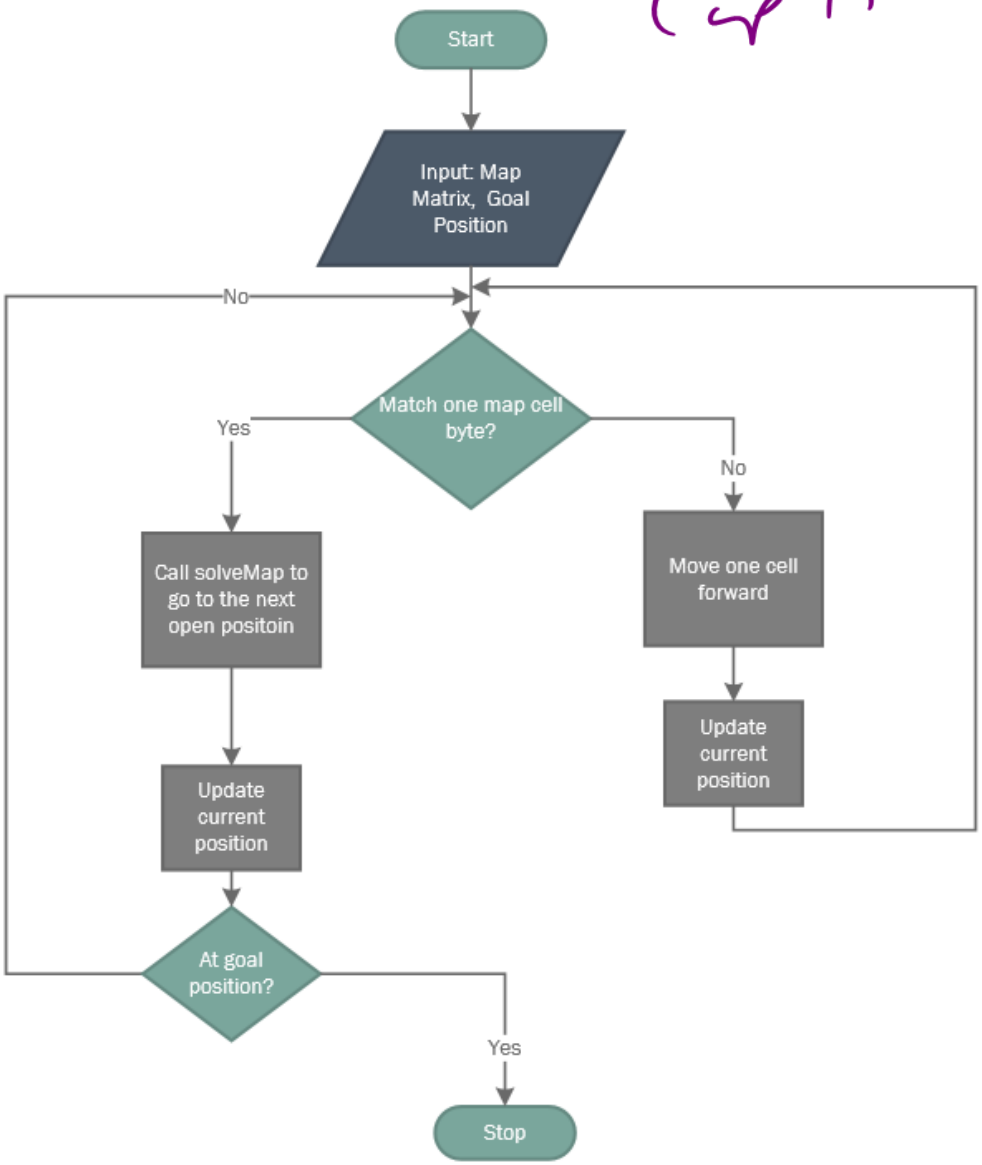
Appendix

A1. Flowchart

Metric Path Planning:

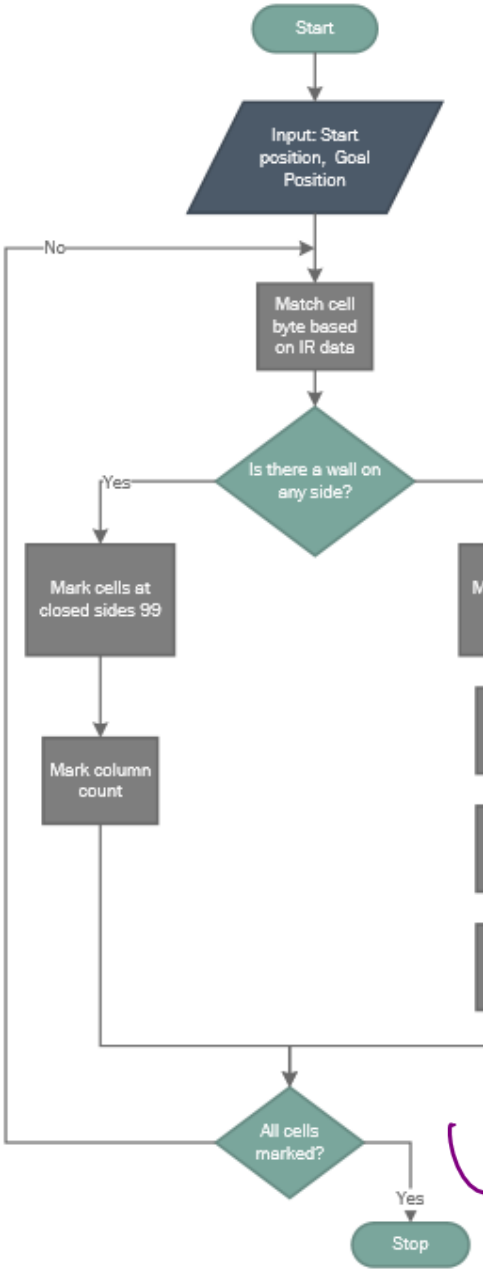


Localization:


Start
Input: Map Matrix, Goal Position
No
Match one map cell byte?
Yes
No
Call solveMap to go to the next open positoin
Move one cell forward
Update current position
Update current position
At goal position?
Yes
Stop

Mapping:

title? figure # caption



Show subsumption architecture